**Practical No 1**

**A) Program to find prime no between 1 to N**

**Aim: Write a program to find prime no between 1 to N in java.**

**Description:**

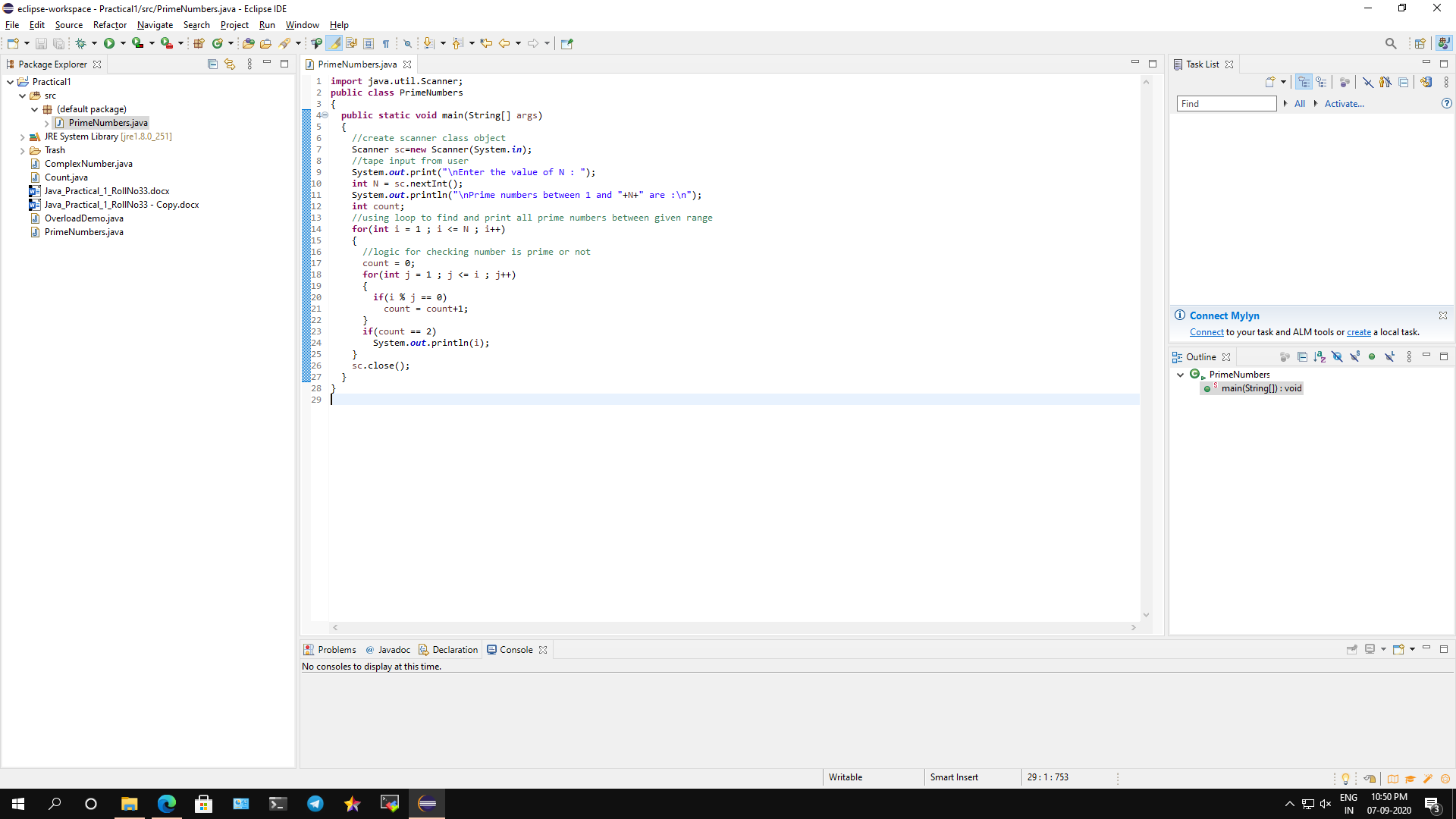
Java is a high-level programming language originally developed by Sun Microsystems and released in 1995. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. Java is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Software Development Domain. Some of the key advantages of learning Java Programming:

* **Object Oriented**: In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
* **Platform Independent**: Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into a platform specific machine, rather into platform independent byte code.
* **Simple**: Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
* **Secure**: With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
* **Architecture-neutral**:  Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of a Java runtime system.
* **Portable**: Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java are written in ANSI C with a clean portability boundary, which is a POSIX subset.
* **Robust**: Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.

We have defined the packages that are imported in the program which is the Scanner util packages. A package in java is used to group related classes. We use packages to avoid name conflicts, and to write a better maintainable code. After the packages we have defined the class name which is class PrimeNumbers. A class is a blueprint for the object, it represents a set of properties or methods that are common to all objects of one type. Now we have created a Scanner class object and then we have defined the main method to accept an array of string arguments. Then it will take the input from the users when we enter the value of N. Once the value is entered it will go into loop till, we find the prime numbers until the value N. It will use logic for checking the prime number till the N value and once we find the prime number the program will stop compiling and get the output.

Some of the Keywords used are:

* **Import:** import is a Java keyword. It declares a Java class to use in the code below the import statement.
* **Class:** A class is a blueprint for the object, it represents a set of properties or methods that are common to all objects of one type.
* **Public:** Public is a Java keyword which declares a member's access as public. Public members are visible to all other classes.
* **Static:** In Java, static keyword is mainly used for memory management. It can be used with variables, methods, blocks and nested classes. It is a keyword which is used to share the same variable or method of a given class.
* **Void:** The void keyword specifies that a method should not have a return value.
* **New:** It creates a Java object and allocates memory for it on the heap. new is also used for array creation, as arrays are also objects.
* **System.in:** System.in in java means to take input from the keyboard or user.
* **System.out.print**: System.out.print in java means to print the output to console.
* **System.out.println:** System.out.println in java means to print the output to console on the next line.



**Conclusion: We have written a program to find prime number between 1 to N in java.**

**Code:**

**import** java.util.Scanner;

**public** **class** PrimeNumbers

{

**public** **static** **void** main(String[] args)

{

//create scanner class object

Scanner sc=**new** Scanner(System.***in***);

//take input from user

System.***out***.print("\nEnter the value of N: ");

**int** N = sc.nextInt();

System.***out***.println("\nPrime numbers between 1 and "+N+" are:\n");

**int** count;

//using loop to find and print all prime numbers between given range

**for**(**int** i = 1 ; i <= N ; i++)

{

//logic for checking number is prime or not

count = 0;

**for**(**int** j = 1 ; j <= i ; j++)

{

**if**(i % j == 0)

count = count+1;

}

**if**(count == 2)

System.***out***.println(i);

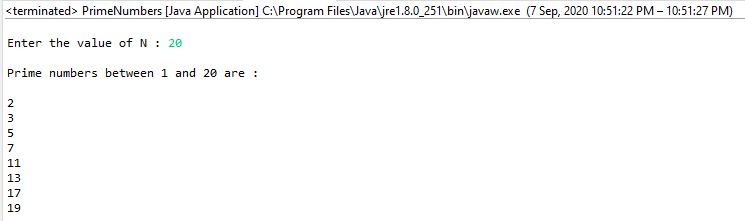
}

sc.close();

}

}

**Output:**



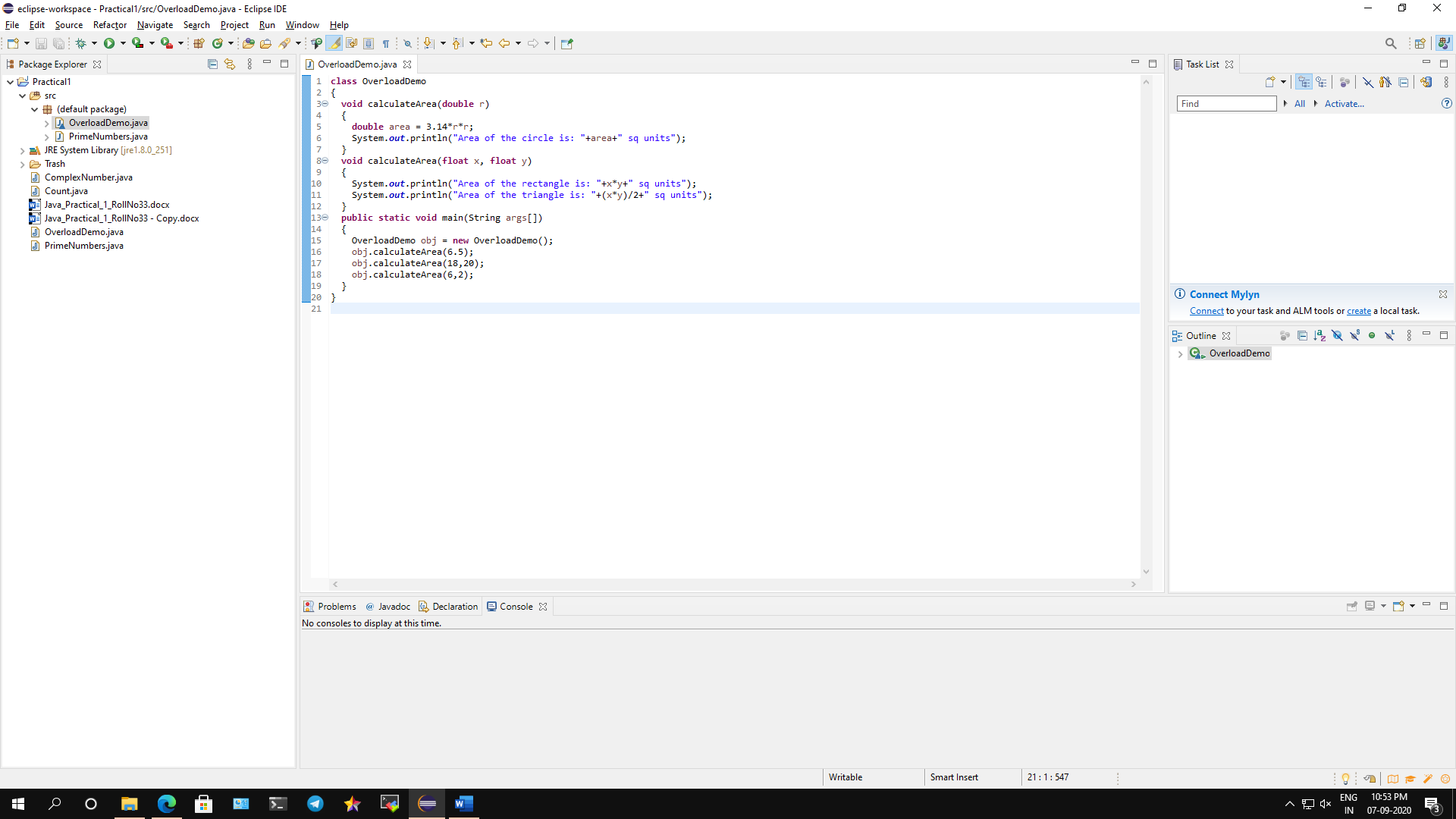
**B) Program to find the function Overloading to find area of circle rectangle and triangle.**

**Aim: Write a java program to find the function Overloading to find area of circle rectangle and triangle.**

**Description:**

We can have two or more methods in java to have the same names with different parameters. These methods are called as method overloading. It is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to [constructor overloading](https://beginnersbook.com/2013/05/constructor-overloading/) in Java.

In the given code we have taken three methods of same name but with different number of arguments with different data types. All the three methods have the same name with different return types, which is much overloading. We also passed different parameters of single, integer and double to calculate the area of circle, rectangle and triangle.



**Conclusion: We have written a program on function Overloading to find area of circle rectangle and triangle.**

**Code:**

**class** OverloadDemo

{

**void** calculateArea(**double** r)

{

**double** area = 3.14\*r\*r;

System.***out***.println("Area of the circle is: "+area+" sq units");

}

**void** calculateArea(**float** x, **float** y)

{

System.***out***.println("Area of the rectangle is: "+x\*y+" sq units");

System.***out***.println("Area of the triangle is: "+(x\*y)/2+" sq units");

}

**public** **static** **void** main(String args[])

{

OverloadDemo obj = **new** OverloadDemo();

obj.calculateArea(6.5);

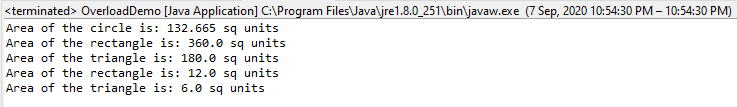
obj.calculateArea(18,20);

obj.calculateArea(6,2);

}

}

**Output:**



**C) Program to implement a program to perform complex number addition, multiplication, subtraction using constructors overloading.**

**Aim: Write a java program to implement a program to perform complex number addition, multiplication, subtraction using constructors overloading.**

**Description:**

We can overload a constructor in java which can be done using constructor overloading.

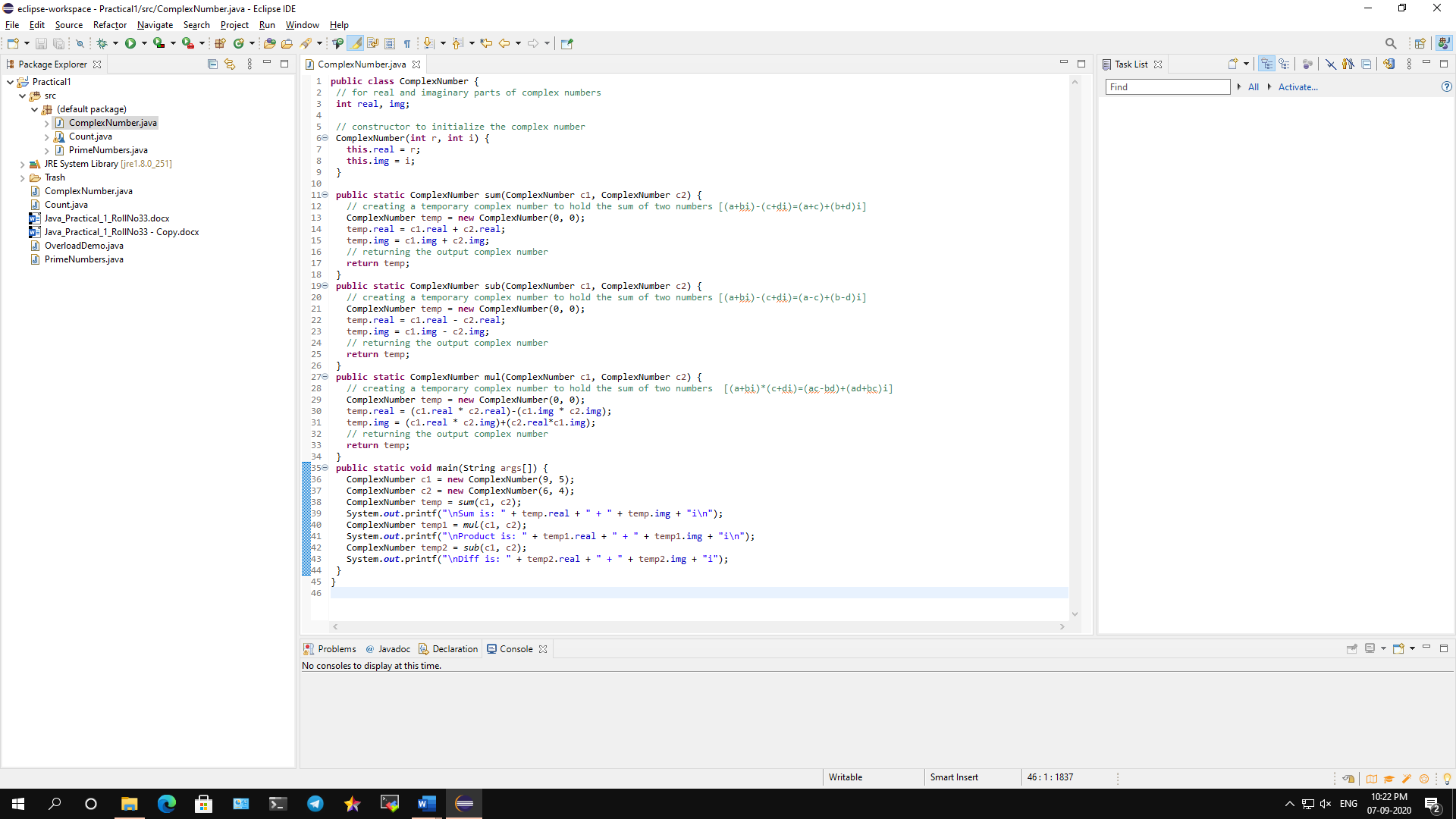
Constructor overloading is a concept of having more than one constructor with different parameters list, in such a way that each constructor performs a different task. There are two types of constructors they are Default constructor and Parameterized constructor.

Following rules must be used for creating java constructor:

- Constructor name must be the same as its class name

- A Constructor must have no explicit return type

In the given code we have created a parameter constructor where 2 arguments can pass data type integers. We then created 3 methods for addition, subtraction and multiplication to find the complex numbers. Then the rest of the required operation will be performed in the main method, as these methods will return the values which are passed to another object of class.



**Conclusion: We have implemented a program to perform complex number addition, multiplication, subtraction using constructors overloading.**

**Code:**

**public** **class** ComplexNumber {

// for real and imaginary parts of complex numbers

**int** real, img;

// constructor to initialize the complex number

ComplexNumber(**int** r, **int** i) {

**this**.real = r;

**this**.img = i;

}

**public** **static** ComplexNumber sum(ComplexNumber c1, ComplexNumber c2) {

// creating a temporary complex number to hold the sum of two numbers [(a+bi)-(c+di)=(a+c)+(b+d)i]

ComplexNumber temp = **new** ComplexNumber(0, 0);

temp.real = c1.real + c2.real;

temp.img = c1.img + c2.img;

// returning the output complex number

**return** temp;

}

**public** **static** ComplexNumber sub(ComplexNumber c1, ComplexNumber c2) {

// creating a temporary complex number to hold the sum of two numbers [(a+bi)-(c+di)=(a-c)+(b-d)i]

ComplexNumber temp = **new** ComplexNumber(0, 0);

temp.real = c1.real - c2.real;

temp.img = c1.img - c2.img;

// returning the output complex number

**return** temp;

}

**public** **static** ComplexNumber mul(ComplexNumber c1, ComplexNumber c2) {

// creating a temporary complex number to hold the sum of two numbers [(a+bi)\*(c+di)=(ac-bd)+(ad+bc)i]

ComplexNumber temp = **new** ComplexNumber(0, 0);

temp.real = (c1.real \* c2.real)-(c1.img \* c2.img);

temp.img = (c1.real \* c2.img)+(c2.real\*c1.img);

// returning the output complex number

**return** temp;

}

**public** **static** **void** main(String args[]) {

ComplexNumber c1 = **new** ComplexNumber(9, 5);

ComplexNumber c2 = **new** ComplexNumber(6, 4);

ComplexNumber temp = *sum*(c1, c2);

System.***out***.printf("\nSum is: " + temp.real + " + " + temp.img + "i\n");

ComplexNumber temp1 = *mul*(c1, c2);

System.***out***.printf("\nProduct is: " + temp1.real + " + " + temp1.img + "i\n");

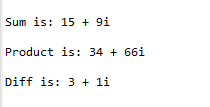
ComplexNumber temp2 = *sub*(c1, c2);

System.***out***.printf("\nDiff is: " + temp2.real + " + " + temp2.img + "i");

}

}

**Output:**



**D) Program to count no of object Created using Static method.**

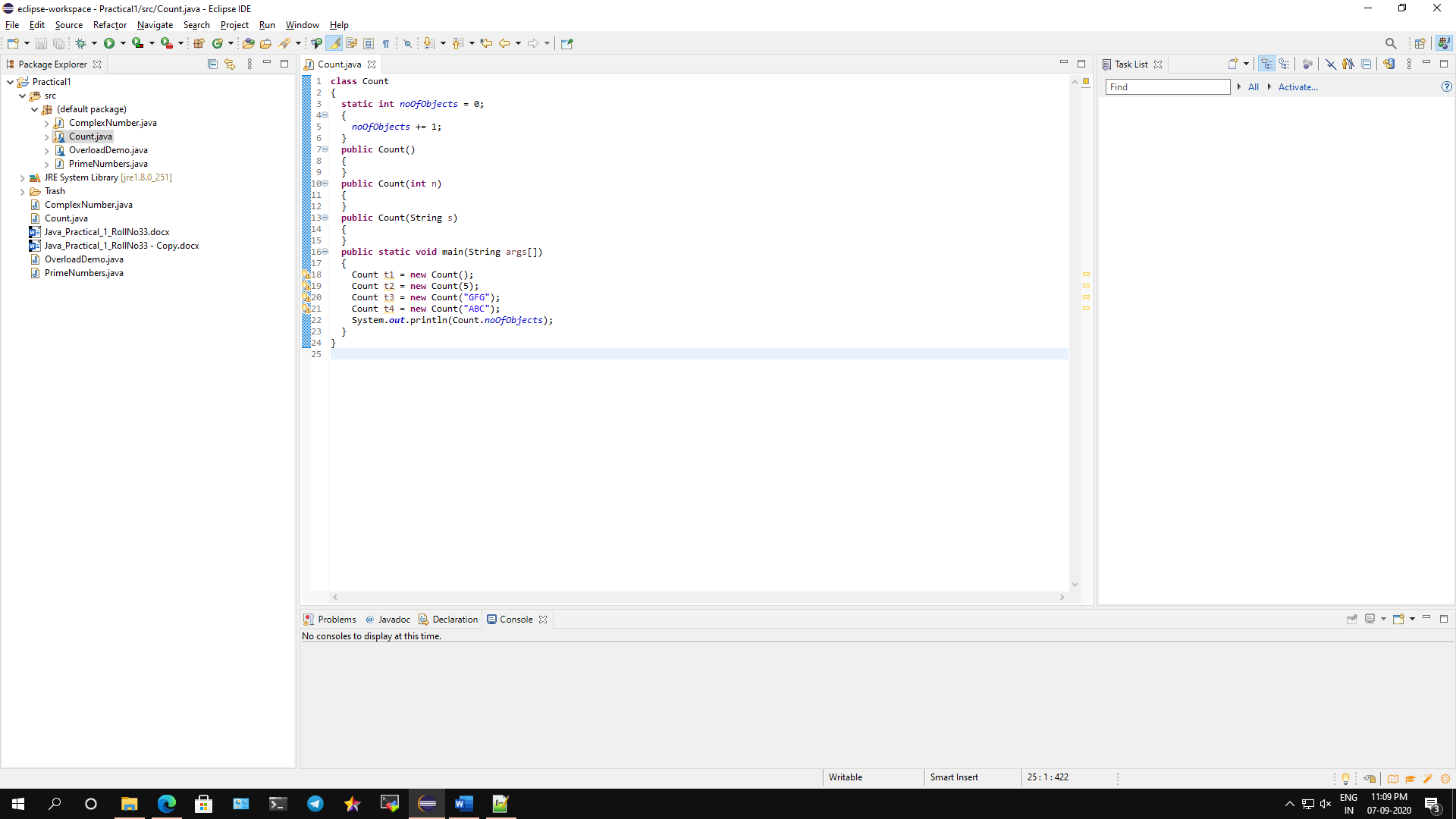
**Aim: Write a java program to count no of object Created using**

**Static method.**

**Description:**

In Java, a static method is a method that belongs to a class rather than an instance of a class. The method is accessible to every instance of a class, but methods defined in an instance are only able to be accessed by that member of a class.

In the given program we have initialized the counter value noOfObjects with 0. This type of variable is static. Then we have defined the public class count. Now in the main static method we have all 4 constructors which are called as 4 objects of the same class. Each time when an object is created the counter value noOfObjects increments to 1. So, the objects were created 4 times and we got the counter value as 4.



**Conclusion: We have written a program to count no of object of object Created using Static method.**

**Code:**

**class** Count

{

**static** **int** *noOfObjects* = 0;

{

*noOfObjects* += 1;

}

**public** Count()

{

}

**public** Count(**int** n)

{

}

**public** Count(String s)

{

}

**public** **static** **void** main(String args[])

{

Count t1 = **new** Count();

Count t2 = **new** Count(5);

Count t3 = **new** Count("GFG");

Count t4 = **new** Count("ABC");

System.***out***.println("Number of object created: "+*noOfObjects*);

}

}

**Output:**

